ARISTA

INTELLIGENT RRM

AI - DRIVEN RADIO RESOURCE MANAGEMENT

TEAM 24

CONTENT

- Problem Understanding
- Additional-Radio Sensing & Scheduling
- Client-View Acquisition
- Multi-Timescale Control Loop
- AI Methods & Data
- Results

PROBLEM UNDERSTANDING

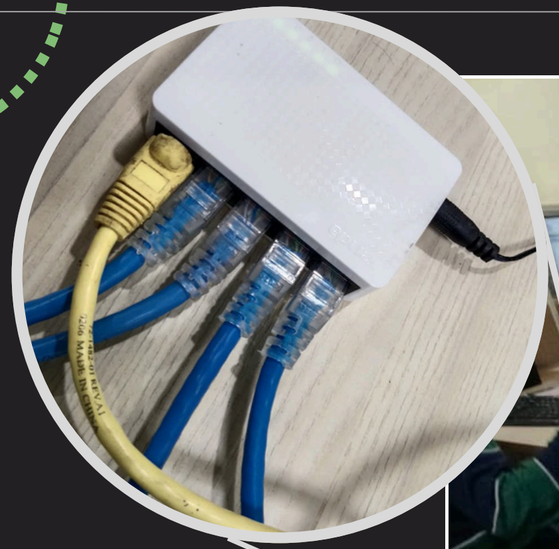
To design an intelligent, multi-radio, learning-driven RRM system for continuous sensing and adaptive, stable Wi-Fi optimization.

Key Challenges

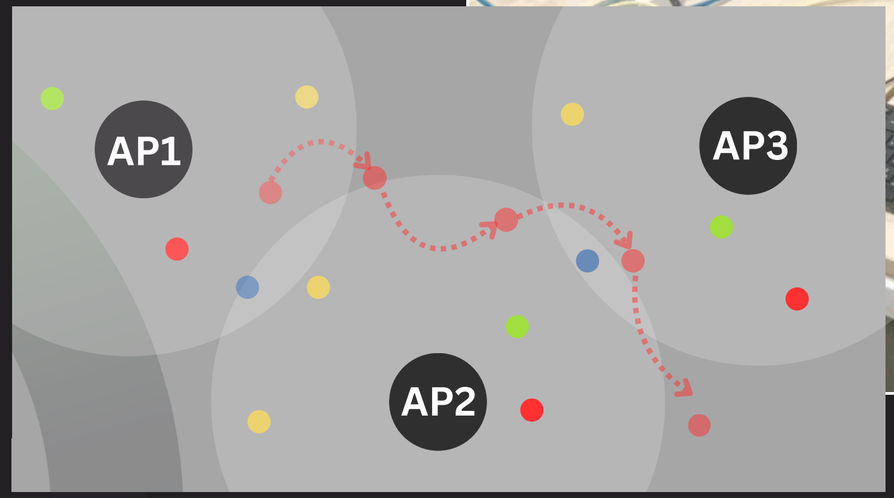
- Client diversity.
- Dynamic channel sensing
- Short-duty, non-Wi-Fi interferers.
- Evolving PHY/MAC features.
- Stability vs agility.

PROPOSED SOLUTION

5 port switch for Data Scheduling between APs and Central Controller



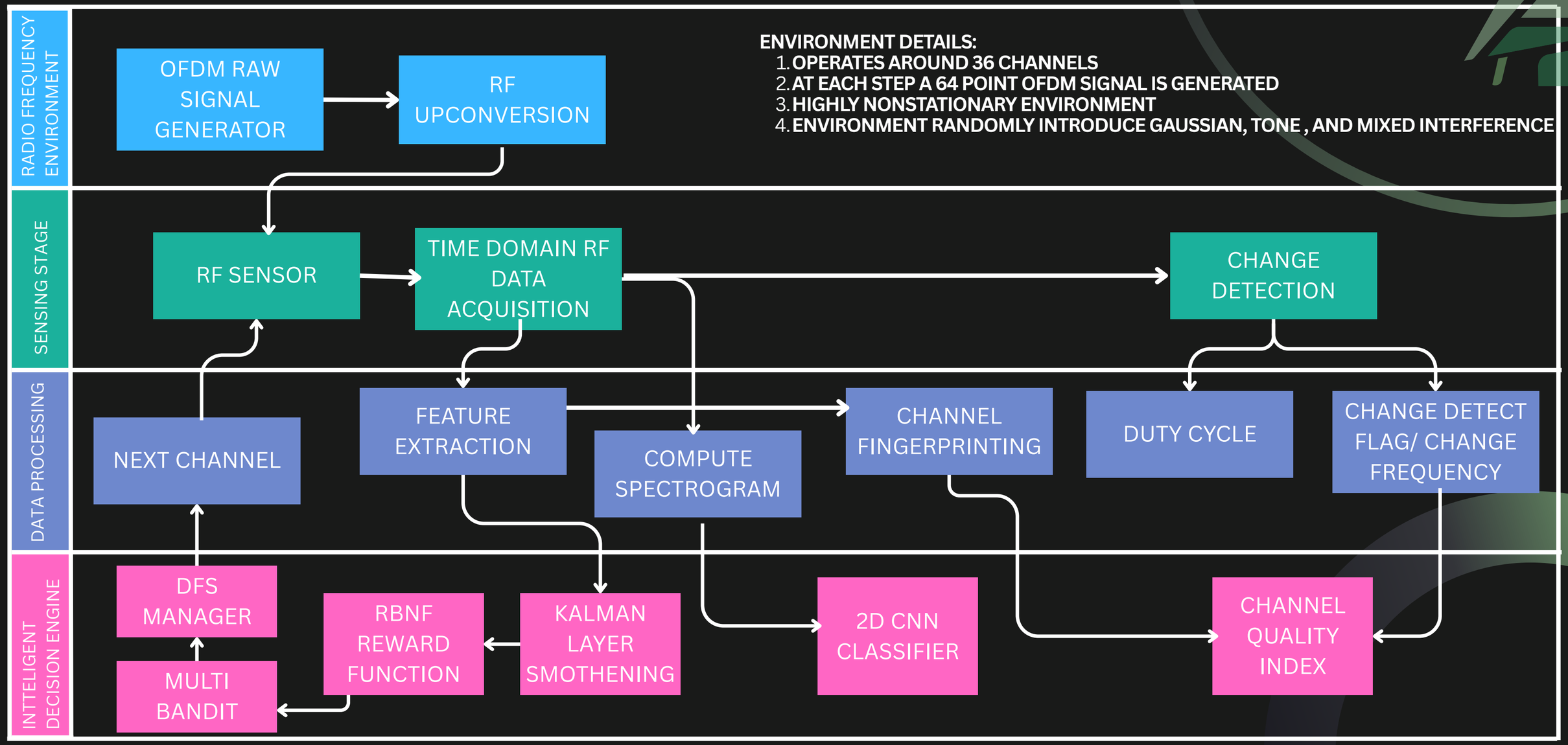
Entire Experimental setup equipments



Experimental Condition

Roaming Clients
Windows Client
IOT Clients
AndroidClient
Edge Clients

SENSING ORCHESTRATOR



FEATURE SELECTION

Measures randomness and uniformity of the spectrum.

$$H_{norm} = \frac{-\sum_k \frac{P[k]}{\sum_i P[i]} \log_2(p_k)}{\log_2(N)}$$

Tells how many frequency bins carry most of the energy.
Find the smallest k where:

$$\sum_{i=1}^{k} P_{sorted}[i] \geq 0.9 \cdot \sum_{i=1}^{N} P[i]$$

And calculate k/N

· High kurtosis means impulsive interference and Low kurtosis means smooth noise or signal.

$$Kurtosis = \frac{E[(x-\mu)^4]}{(E[(x-\mu)^2])^2}$$

Measures asymmetry of amplitude distribution.

$$Skewness = \frac{E[(x-\mu)^3]}{(E[(x-\mu)^2])^{3/2}}$$

A robust alternative to standard deviation — ignores outliers.

$$Median = median(x)$$
$$MAD = median(|x - Median|)$$

Estimates noise variance by checking mismatch between the cyclic prefix and end of each OFDM symbol.

$$J(u) = \frac{1}{2M(N_g-(u-1))} \sum_{m=1}^{M} \sum_{k=u}^{N_g} |y_m[N+k] - y_m[N_g-k]|^2$$

$$\sigma_{CP}^2 = \min_u J(u)$$

Average correlation between the signal and its lagged version.

$$Autocorr\ Energy = \frac{1}{L} \sum_{l=1}^{L} \left| \frac{(x_{0:N-l} - \bar{x}_1) \cdot (x_{l:N} - \bar{x}_2)}{||x_{0:N-l} - \bar{x}_1|| \cdot ||x_{l:N} - \bar{x}_2||} \right|$$

Measures the residual power left after estimating the channel using known pilot tones.

$$\hat{H}_p = \frac{Y_p}{X_p}, Residual = Y_p - \hat{H}_p X_p$$

$$\sigma_{pilot}^2 = E[|Residual|^2]$$

Estimates average subband noise level and captures uneven or frequency-selective noise.

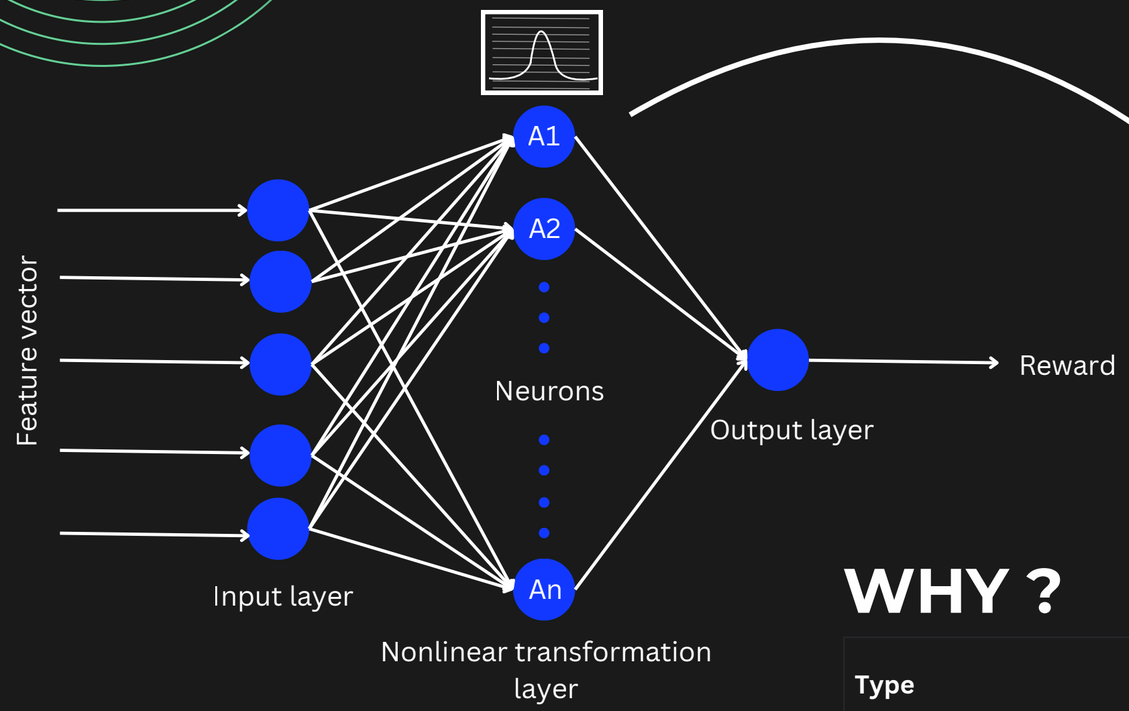
$$P_{sub,i} = \sum_{k \in band_i} |S[k]|^2.$$

$$\bar{P}_{noise} = \frac{1}{N_b} \sum_i P_{noise,i}$$

RBNF REWARD FUNCTION

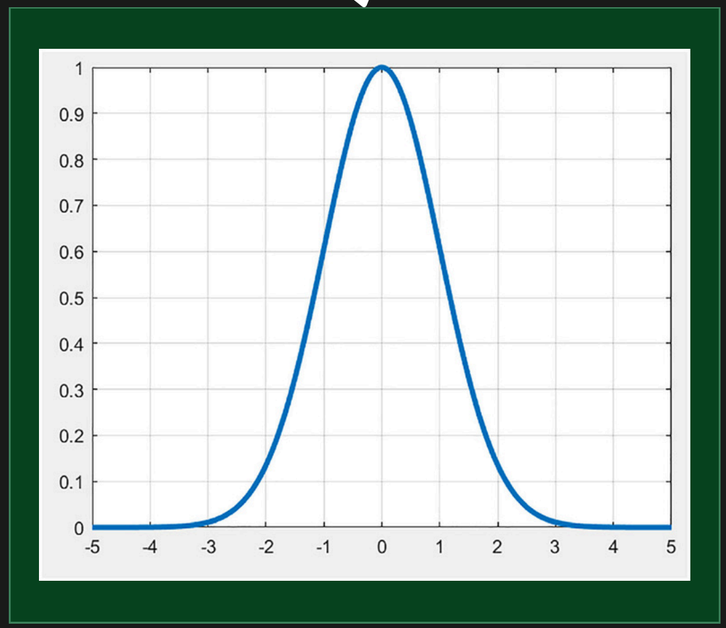
The RBFN is tuned using Bayesian Optimization, making the reward function adaptive to the radio environment without manual parameter selection

RADIAL BASIS FUNCTON



$$\phi(r) = exp\left(\frac{-r}{\sigma}\right)^2$$

r represents the Euclidean distance between the input vector(x) and the function's center point (c), defined as r =||x-c||



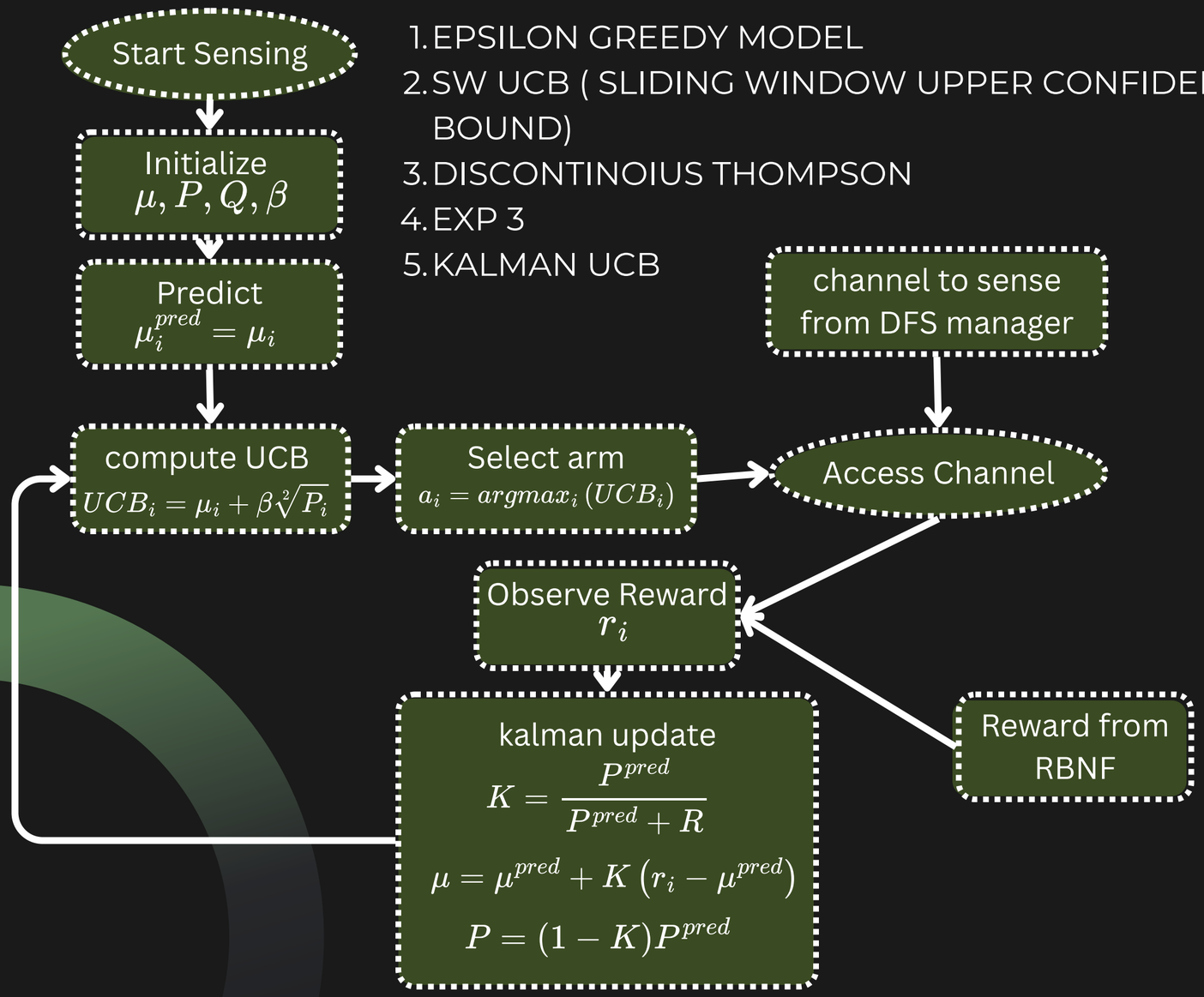
WHY ?

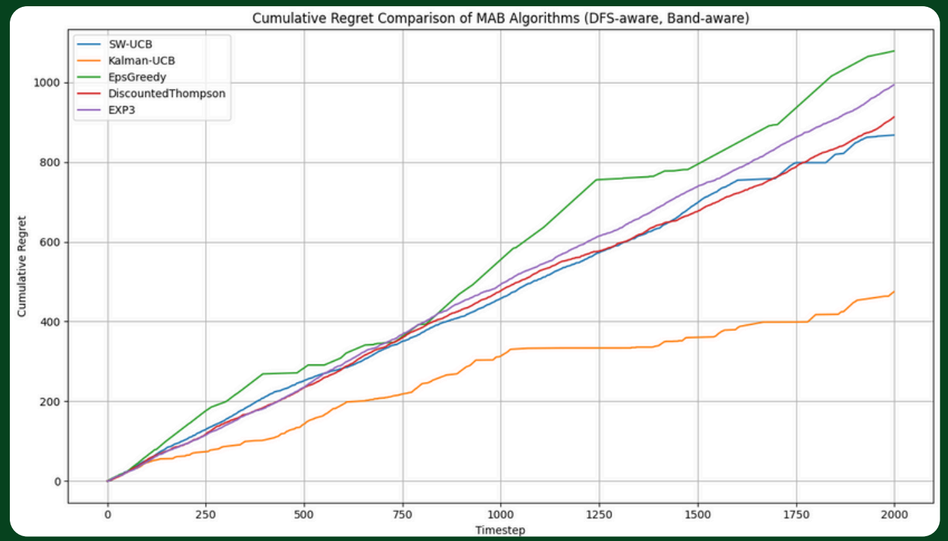
It is difficult to make a combined Reward Function as some parameter is good for detecting some specific type of noise to, we make an approach of training the reward function using Radial Basis Network to make the Reward more vertesile for different kind of noise.

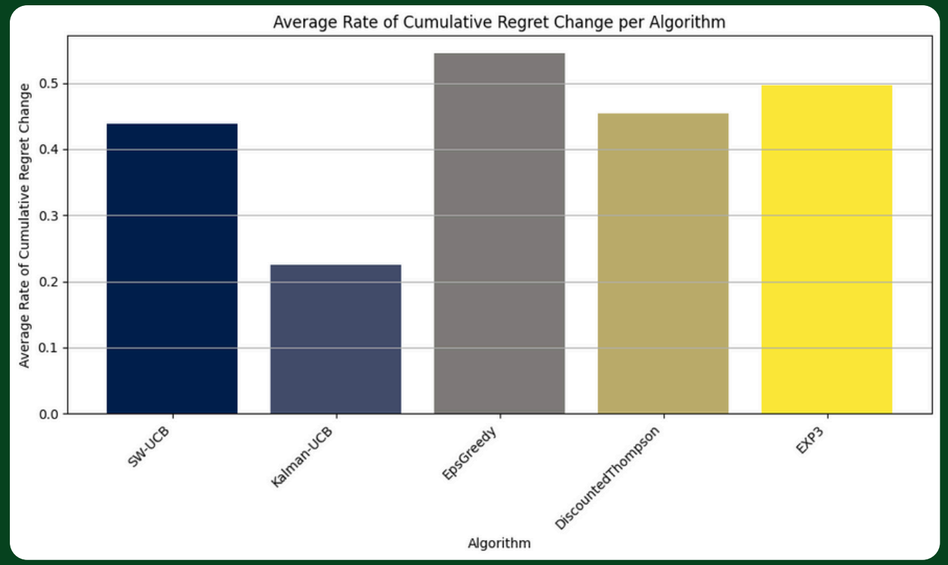
Type	Example Features	Detects
Time-domain	MAD, RMS, Kurtosis, Skewness, PAR	impulsive noise, hums, DC offsets
Frequency-domain	PSD, Entropy, Flatness, Occupied BW	broadband vs narrowband interference
OFDM aware	CP Sigma, Pilot Noise, Subband Mean	Wi-Fi/BLE/Microwave channel noise characterization

SELECTION OF MULTIBANDIT ALGORITHM

1. EPSILON GREEDY MODEL
2. SW UCB (SLIDING WINDOW UPPER CONFIDENCE BOUND)
3. DISCONTINOIUS THOMPSON
4. EXP 3
5. KALMAN UCB

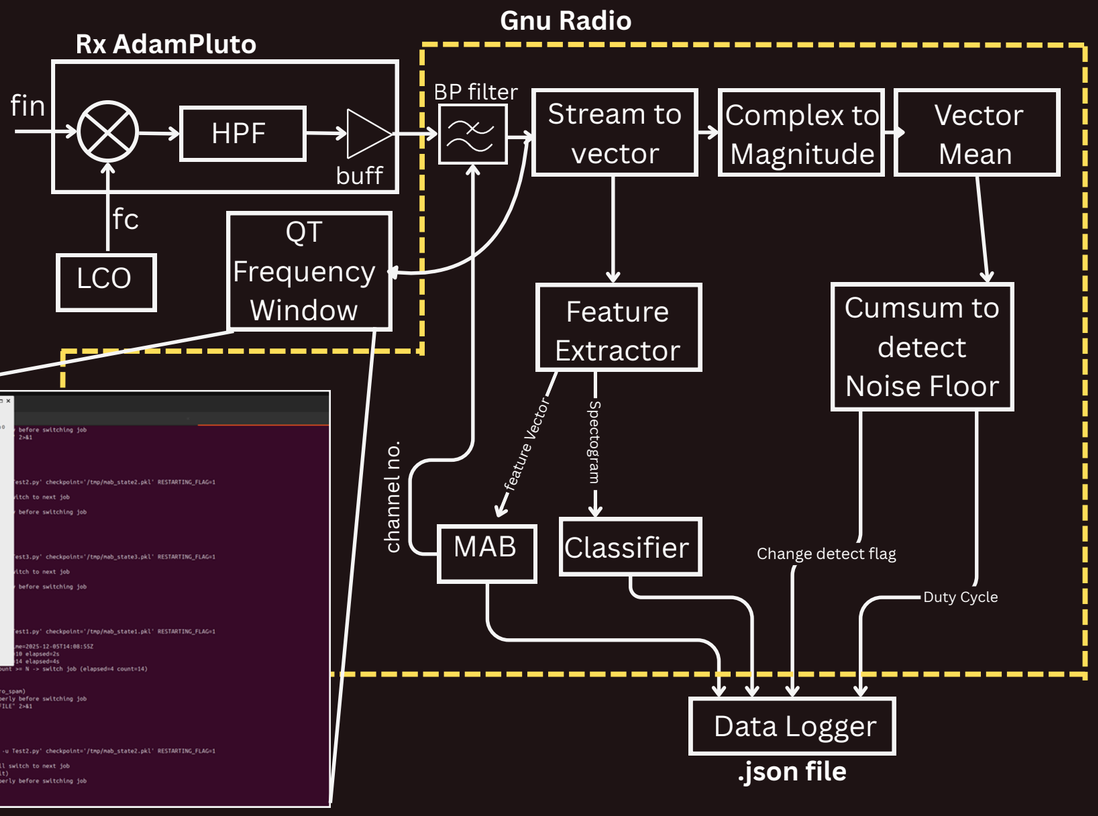


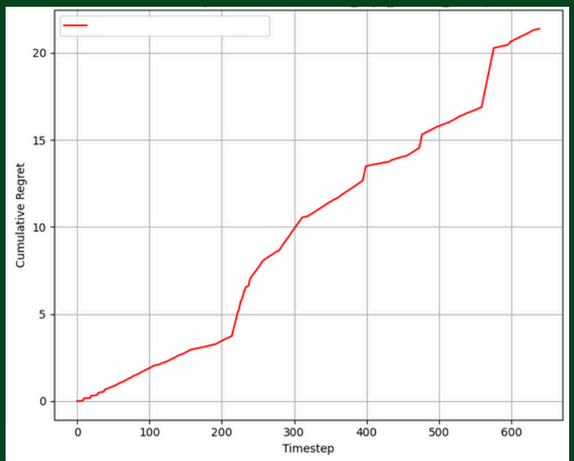




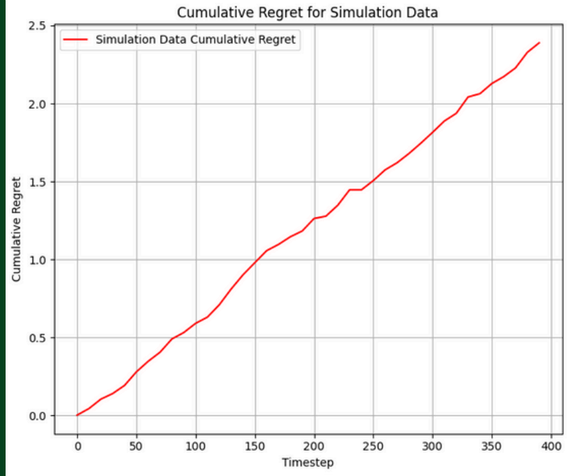
SENSING ORCHESTRATOR DESIGN IN SDR

THE ORCHESTRATOR IS TESTED ON AN HIGHLY UNSTABLE ENVIOURMENY WHERE INTERFERENCE ARE INJECTTED INTO THE ENVIRONMENT CONTINIOUSLY WITH SOME PROBABILITY OF INJECTION.





Channel State	Total	Avg Time	Regret/s	Reward/s
Non-DFS Channel	566	0.0300	1.1104	31.3547
NOP : DFS Channel	18	0.6111	0.0717	1.5239
AVAILABLE : DFS Channel	50	0.2000	0.1490	4.7265
RADAR : DFS Channel	6	0.1667	0.2223	5.6283



Band	Total	Avg Time/ch	Regret/s	Reward/s
2.4GHz	832	0.075636	0.11890	0.02468
5GHz (Non-DFS)	1107	0.123000	0.16940	0.03270
5GHz (DFS)	61	0.003812	0.01280	0.00121

2D CNN INTERFERENCE CLASSIFIER

Data Generation & AI Approach

Robust Data Foundation:

- 92k Synthetic MATLAB samples.
- 5 Classes: WiFi, BT, ZigBee, Microwave, CW.
- Inputs: 2D Spectrograms + 1D Statistical properties.

Architectural Search:

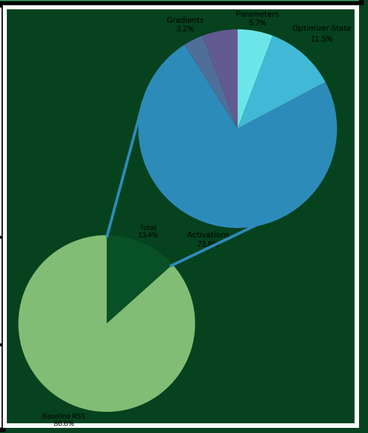
- Baselines: ResNet, ImageNet, ConvNext (discarded due to size).
- Optimization: Scaled down to EfficientNet family for edge deployment.

Winning Model: EfficientNet-Lite 1

- Superior robustness on spectrogram data.
- Performance: Macro-F1 ≈ 0.99

RAM BUDGET SUMMARY

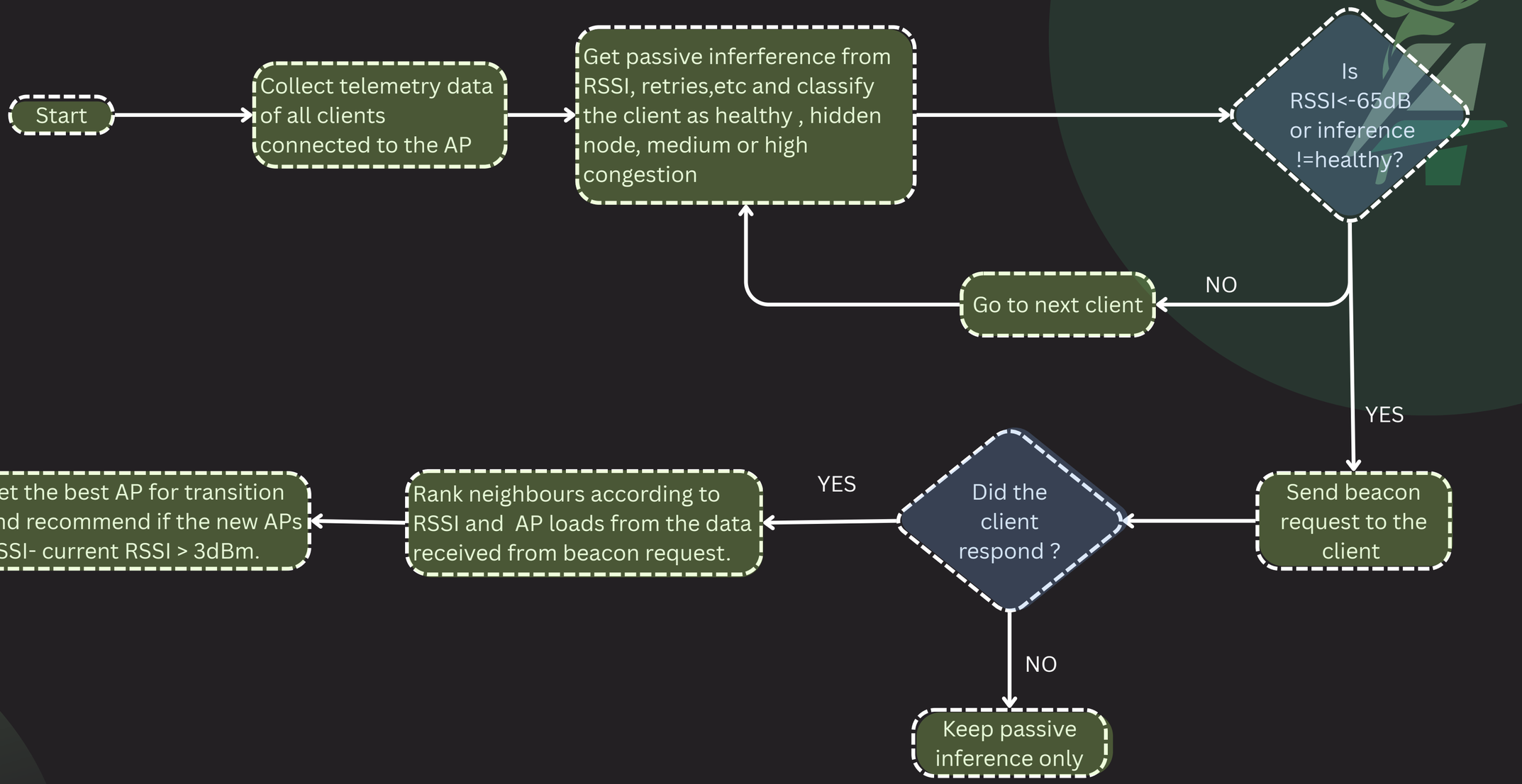
Parameters		16.43 MB
Optimizer State		32.86 MB
Activations		211.00 MB
Gragients		9.19 MB
DataLoader	+	16.43 MB
Total		285.91 MB
Baseline RSS	+	1.81 GB
Final Budget		2.09 GB



CLASSIFIER PERFORMANCE

Class	Precision	Recall	F-1 score	Support
BLE	0.99	0.97	0.98	6311
ZIGBEE	1.00	0.99	1.00	6448
CW	1.00	0.99	0.99	6451
FHSS	1.00	1.00	1.00	6380
MICROWAVE	0.99	0.96	0.97	6342
NONE	0.98	1.00	0.99	5999
Micro avg	0.99	0.99	0.99	37931
Macro avg	0.99	0.99	0.99	37931
Weighted avg	0.99	0.99	0.99	37931

CLIENT VIEW ACQUISITION AND BSS TRANSITION RECOMMENDATION



CLIENT VIEW ACQUISITION

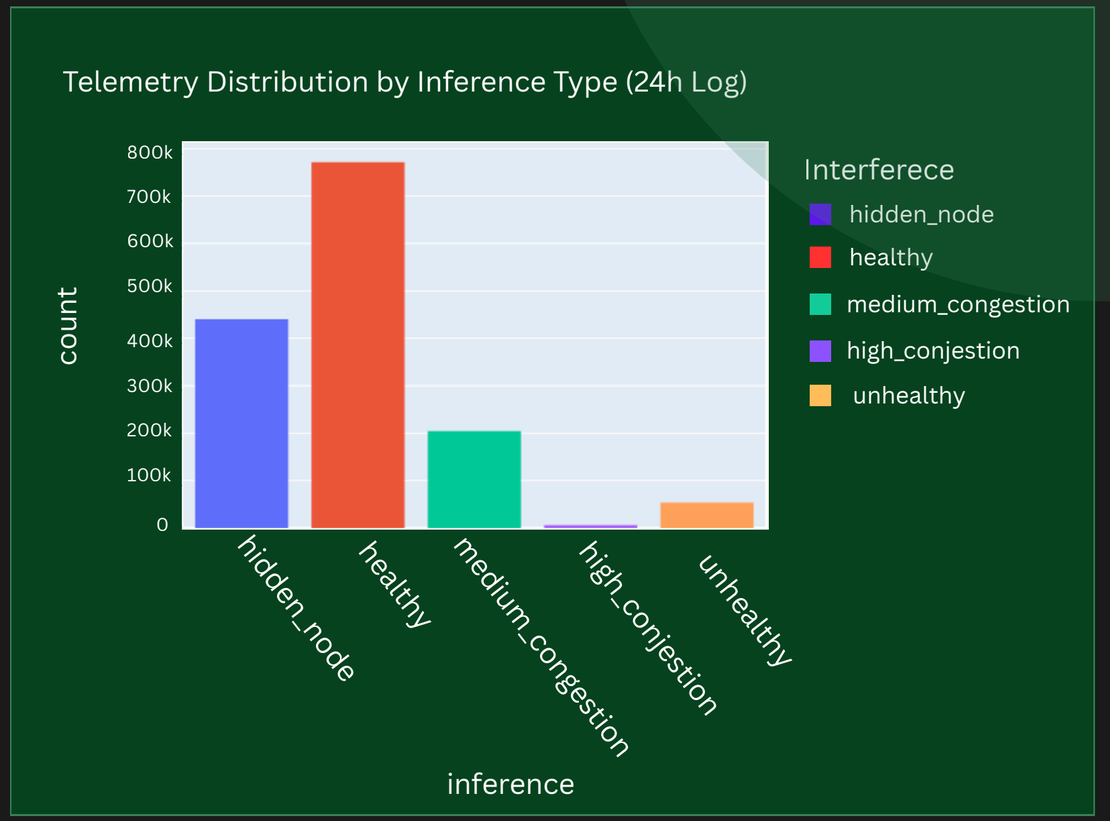
TELEMETRY SCHEMA

Field	Description
timestamp	Time at which data is received
ap, ap_channel, channel_width,transmit power	Serving AP name and operating channel.
ap_load	Number of clients connected to each AP
client hashed mac address	MAC Address of clients with hash function applied
rssi, mcs	RSSI (dBm), SNR (dB) and MCS index (if available)
retry_up, retry_down	Uplink / downlink retry counts
throughput_mbps	Data rate
inference	Health label: healthy, hidden_node, medium/high_congestion, unhealthy, un known.
neighbors	Optional list of neighbour AP reports (RSSI, channel, load)

Data Source	Telemetry Data
Access Point	AP channel,Channel width,Transmit power
Client View	Client Hashed Mac Address,RSSI,MCS,Retry Down,Retry Up,Throughput Mbps and Neighbour APs data

ACCEPTANCE METRICS

OS	BSS Transition attemps	Accepted	Acceptance rate	QoE Gain	RSSI Gain
macOS	556	522	0.939	0.1026	4.687
Windows	368	337	0.916	0.1130	5.020
Android	312	218	0.699	0.1120	4.987
Linux	335	308	0.919	0.1010	4.917



Advanced Client View: Passive RTT Analysis Framework

RTT, RSSI Binning, CUSUM Anomaly Detection

Clients (Generating TCP Packets)

- Works without 802.11mc
- Fully passive, real-time RTT analysis

Packets (with TSval)

Packet Capture Layer (Extract TSval & TSecr)

Timestamp Pairs

RTT Computation Engine
Copute RTT from TSval ⇔ TSecr correlation

Per-packet RTT

Metric Aggregator
RTT Median
RTT P95
Loss rate
Loss Variance

Time-windowed metrics

CUSUM Anomaly Detector
Detects RTT spikes

Display

Output

RSSI-Based Client Binning

Near	Mid	Edge
> −45 dBm	−45 to −65 dBm	less than -65 dBm

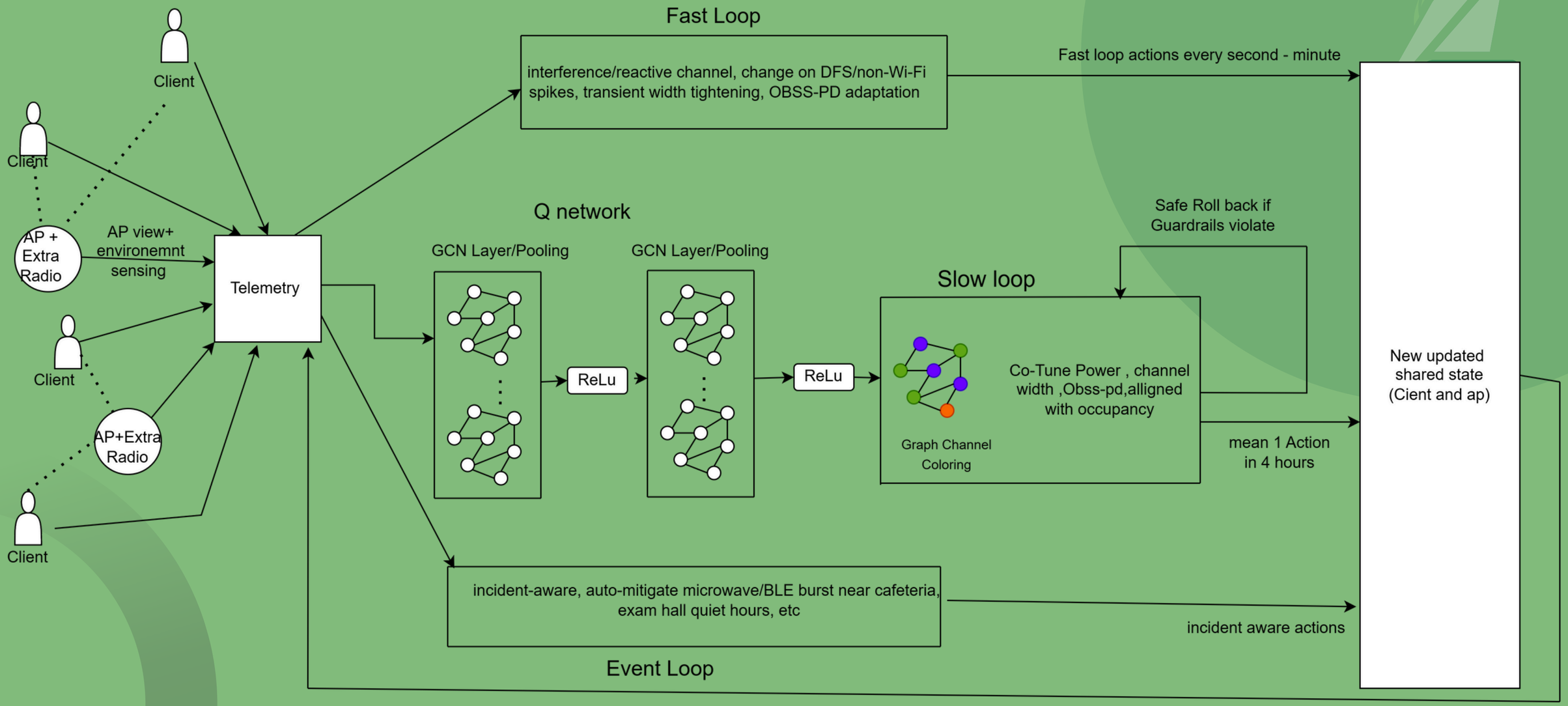
- **Approximates user spatial location**
- **Enhances QoE + performance correlation**
- **Replaces 802.11mc-based ranging**

Some RTT Results

Samples : 50
RTT Median : 0.26 ms
RTT P95 : 1.62 ms
Loss rate : 3.85 %
Loss variance : 0.036982

Samples : 50
RTT Median : 0.33 ms
RTT P95 : 4.20 ms
Loss rate : 1.96 %
Loss variance : 0.019223

MULTI-TIMESCALE CONTROL LOOPS



AI METHODS | INTERFERENCE GRAPH

The interference graph is built by calculating the RSSI values for AP-AP relations using the formulas above,as the distance between two APs is known.

$$\text{RSSI} = \frac{P_t \cdot G_t \cdot G_r \cdot \lambda^2}{(4\pi)^2 \cdot d^2 \cdot L} + P_n + P_i^{\text{combined}}$$

$$P_i^{\text{combined}} = \frac{P_{i1} + P_{i2}}{2} \cdot \text{overlap}$$

$$P_{\text{n}} = N_{\text{floor}}(\text{dBm}) = -174 + 10\log_{10}(B) + NF + 10\log_{10}\left(\frac{T}{290}\right) + \Delta NF$$

$$\text{overlap} = \frac{\text{intersect}(f_1, f_2)}{\text{union}(f_1, f_2)}$$

```
AP_NOISE_FIGURES_DB = {
    "ap1": {  # MediaTek MT7921
        "chipset": "MT7921",
        "NF_dB": 4.0,
    },
    "ap2": {  # Realtek RTL8852E
        "chipset": "RTL8852E",
        "NF_dB": 5.0,
    },
    "ap3": {  # Intel AX211
        "chipset": "AX211",
        "NF_dB": 1.5,
    },
}
```

- P_t = Transmit power (in watts)
- G_t, G_r = Antenna gains (linear)
- λ = Wavelength (in meters)
- d = Distance (in meters)
- L = System loss
- P_n = Noise floor (in watts)
- P_i = Interference power (in watts)

INTERFERENCE GRAPH

The function **compute single ap to ap rssi(...)** is then employed. This function models
the Friis transmission equation, scaled by the calculated overlap, and adds the noise floor and P_i_combined .
The result is a comprehensive neighbours list for each AP, structured as:

```
{"ap_id": neighbor_id, "rssi": rssi_ij, "distance_m": d
```

Edges carry specific features essential for the learning model: rssi (from the RF model), dist (meters), and overlap (0–1).

$$w_{ij} = \exp\left(\frac{\text{RSSI}_{ij} + 95}{20}\right) \cdot \Phi(d_{ij}) \cdot \text{overlap}_{ij}$$

$$\Phi(d_{ij}) = \begin{cases} \frac{1}{d_{ij}^2}, & \text{if } d_{ij} > 0 \\ 1, & \text{if } d_{ij} = 0 \end{cases}$$

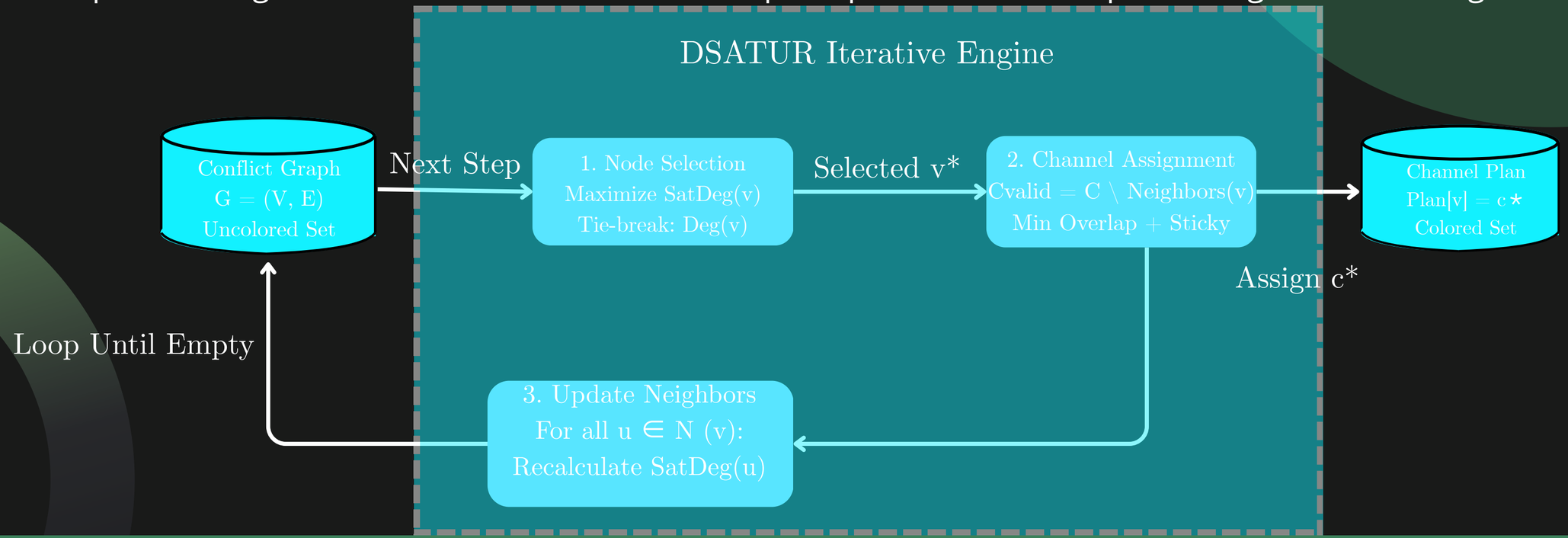
DSATUR Channel Coloring:

Conflict graph tells us which APs (nodes) must not share the same channel if they are strong neighbours i.e AP–AP RSSI above threshold -85.00 dBm ,so if they have same/overlapping channels they will interfere

Saturation Degree (sat_deg) :number of unique channels (colors) currently assigned to an AP's neighbors.

Flowchart Definitions:

- **Minimize Overlap :**Standard graph coloring which is binary but Wi-Fi channels can partially overlap ,select the channels that yields the lowest total spectral overlap.
- **Stickness:** Stability is ensured via stickness i.e. if previous channel is valid not forbidden by neighbour and its interference score is within a acceptable margin of a new best candidate we keep the previous channel preventing channel change for marginal gain.



INTERFERENCE GRAPH

To utilize Deep Learning, the system transforms the snapshot and Grl into a PyTorch Geometric (PyG) graph using the function build pyg graph(snapshot, G rl).

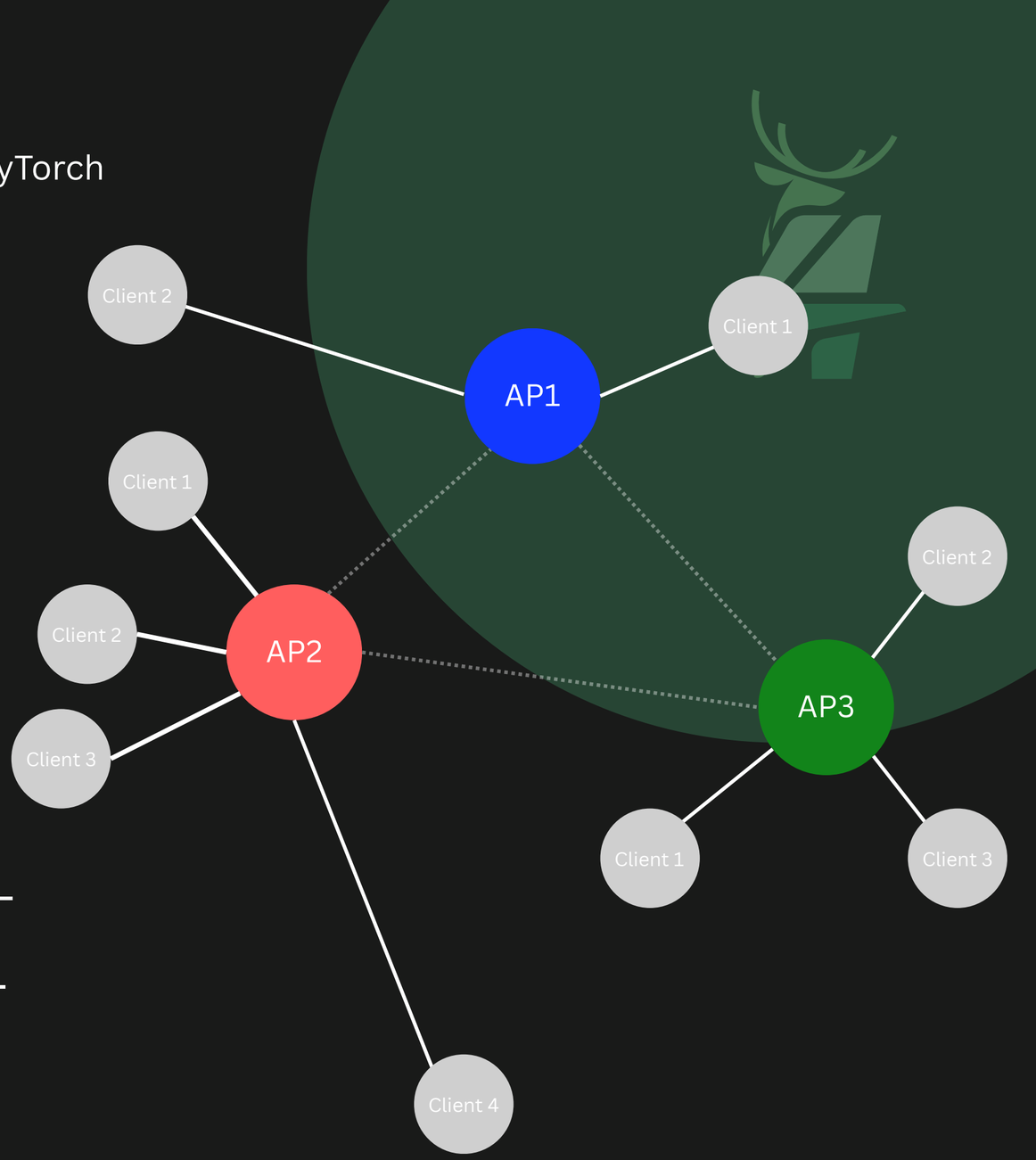
Graph Neural Network Representation:

Node Features:

- Channel,Bandwidth,
- Tx Power,
- OBSS Preamble Detection (obss pd),
- Number of Clients (num clients)
- Minimum Client RSSI,
- Mean Client RSSI
- Mean Tx Retries (tx retries)
- Mean Tx Failed (tx failed)
- Minimum Client Distance (distance graph m)
- Mean Client Distance (distance graph m)

The distance between the AP and client can be derived as:

$$d = \sqrt{\frac{P_t \times G_t \times G_r \times \lambda^2}{(4\pi)^2 \times L \times (RSSI - P_n - P_i)}}$$



Edge Attributes = [weight, rssi , $\frac{1}{dist^2}$, overlap]

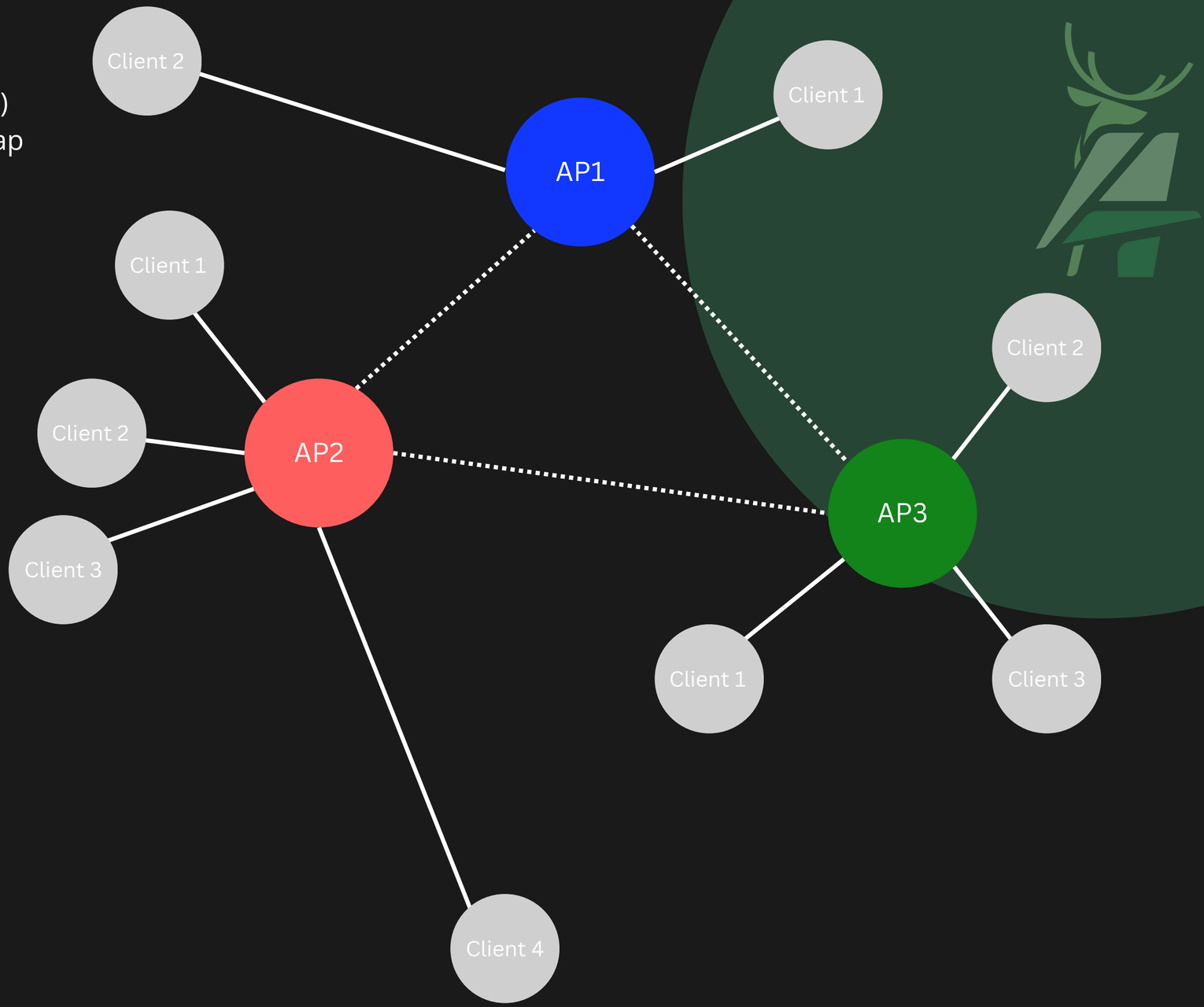
The final output is a Data(x, edge index, edge attr) object fed into the GNNQNetwork,along with an ap index map to map tensor indices back to AP IDs.

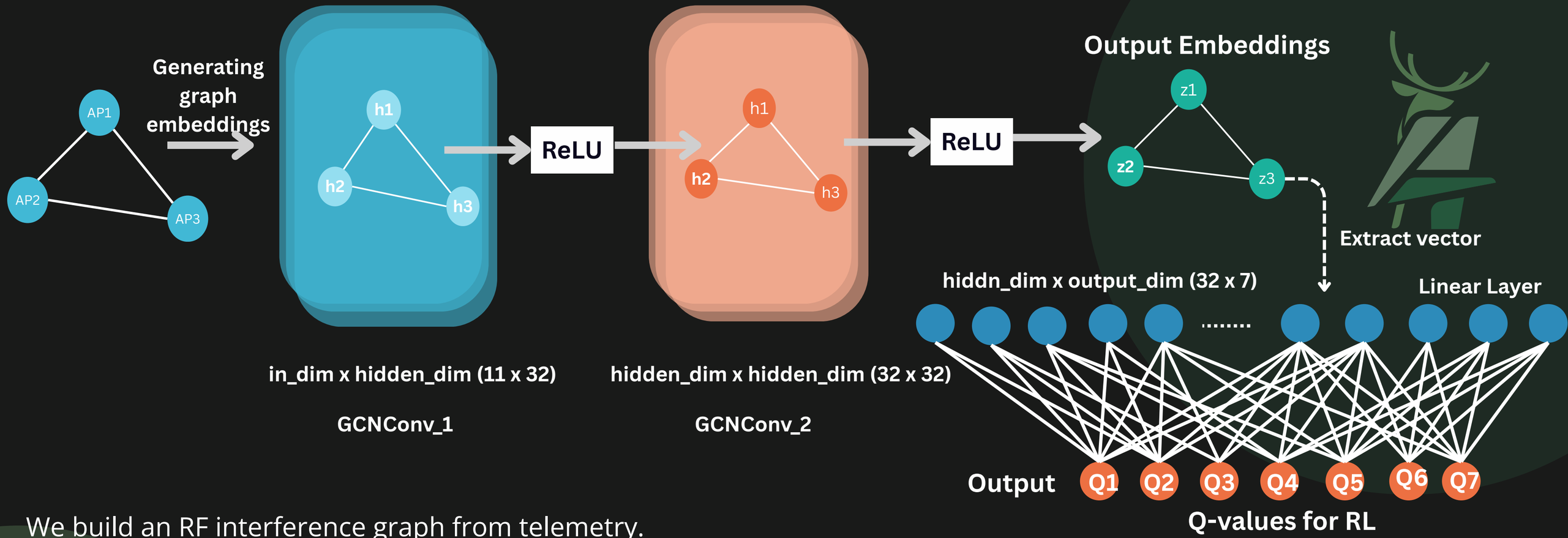
Why we didn't treat clients as nodes ?

The RL graph is meant to control AP-level knobs, not per-client knobs, so the natural node is the AP, with clients only feeding into AP features (min/mean RSSI, retries, distance).

Clients are numerous and short-lived; putting every client as a node would explode graph size, make training unstable, and add a lot of churn for very little extra structure.

Aggregating client info into AP-level statistics gives almost the same signal (edge/near clients, coverage, QoE) while keeping the graph small, stable and easy to debug.





We build an RF interference graph from telemetry.
From this we derive:

- A **conflict graph** for **DSATUR-based channel coloring** (AP nodes only).
- A PyG graph (AP nodes + interference edges + rich node/edge features) that feeds a **GNN Q-network**.
- The GNN Q-network is trained via offline CQL on logs of (graph snapshot, RRM actions, QoE rewards).

This makes the RL policy graph-aware: each AP's action depends not just on its own KPIs but on the state of its RF neighbours, leading to more sensible power/BW/OBSS-PD tuning than a per-AP MLP.

```
# Discrete action space (relative steps) per AP
NO_OP      = 0
POWER_UP   = 1
POWER_DOWN = 2
BW_UP      = 3
BW_DOWN    = 4
OBSS_UP    = 5
OBSS_DOWN  = 6

NUM_ACTIONS = 7

# Discrete grids and step sizes
# Power in dBm
PWR_MIN  = 10.0
PWR_MAX  = 20.0
PWR_STEP = 3.0

# OBSS-PD in dBm
OBSS_MIN  = -82.0
OBSS_MAX  = -62.0
OBSS_STEP = 2.0     # 2 dB granularity

# Bandwidth values in MHz
BW_VALUES = [20, 40, 80]
```

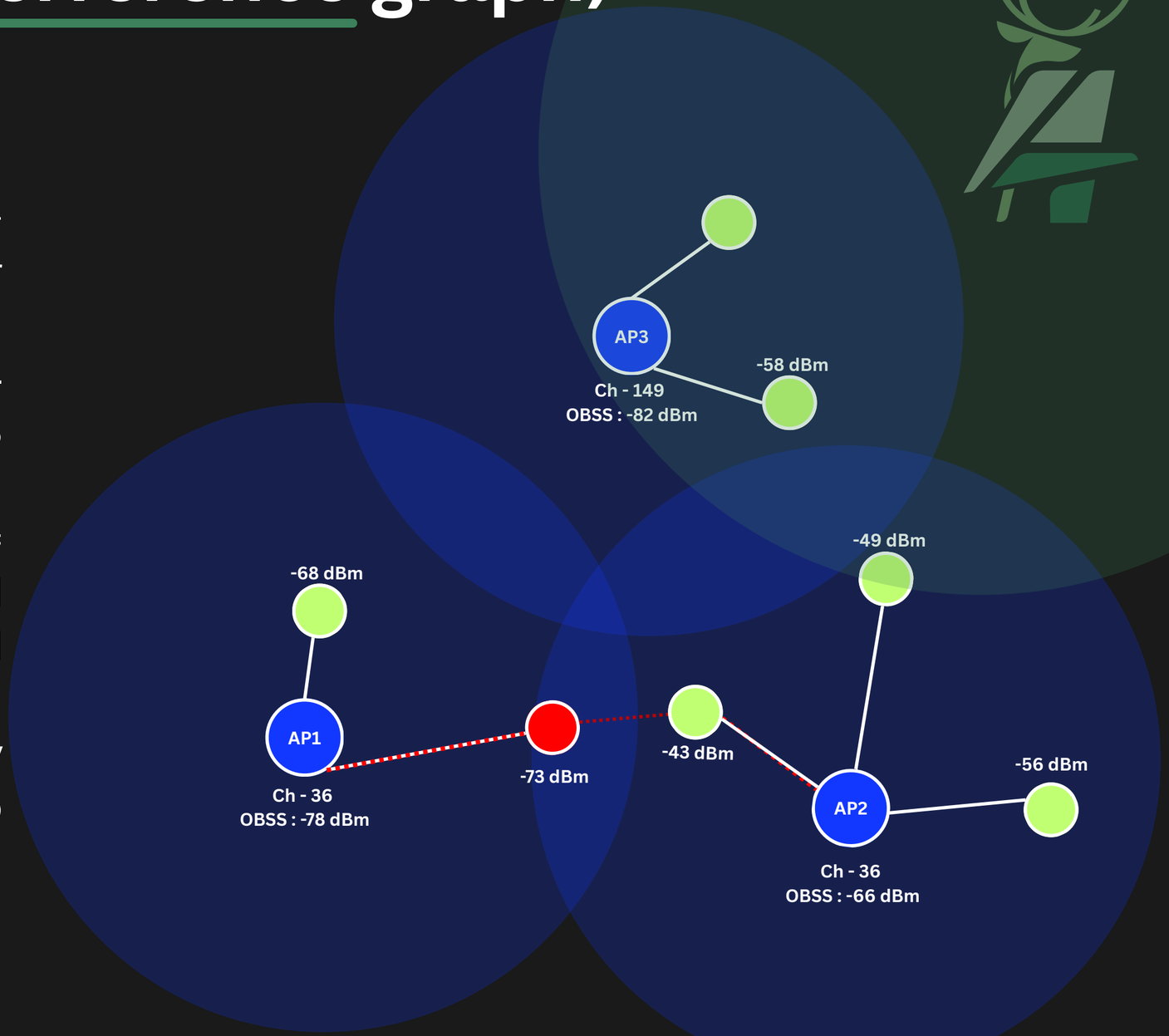
In the given screenshot are the 7 actions of the RL:

Reason for our parameters:

Given only 3 APs in the graph per snapshot, attention weights would be noisy and not worth the extra complexity.We use a small GCN **(GCNConv, hidden dim = 32)** because our interference graph is tiny and data-limited. GCN gives stable neighbour aggregation without the extra parameters and instability of GAT/GraphSAGE, and a 32-dim embedding is enough capacity while avoiding overfitting and keeping the slow-loop controller lightweight

Emulated OBSS-PD (Interference graph)

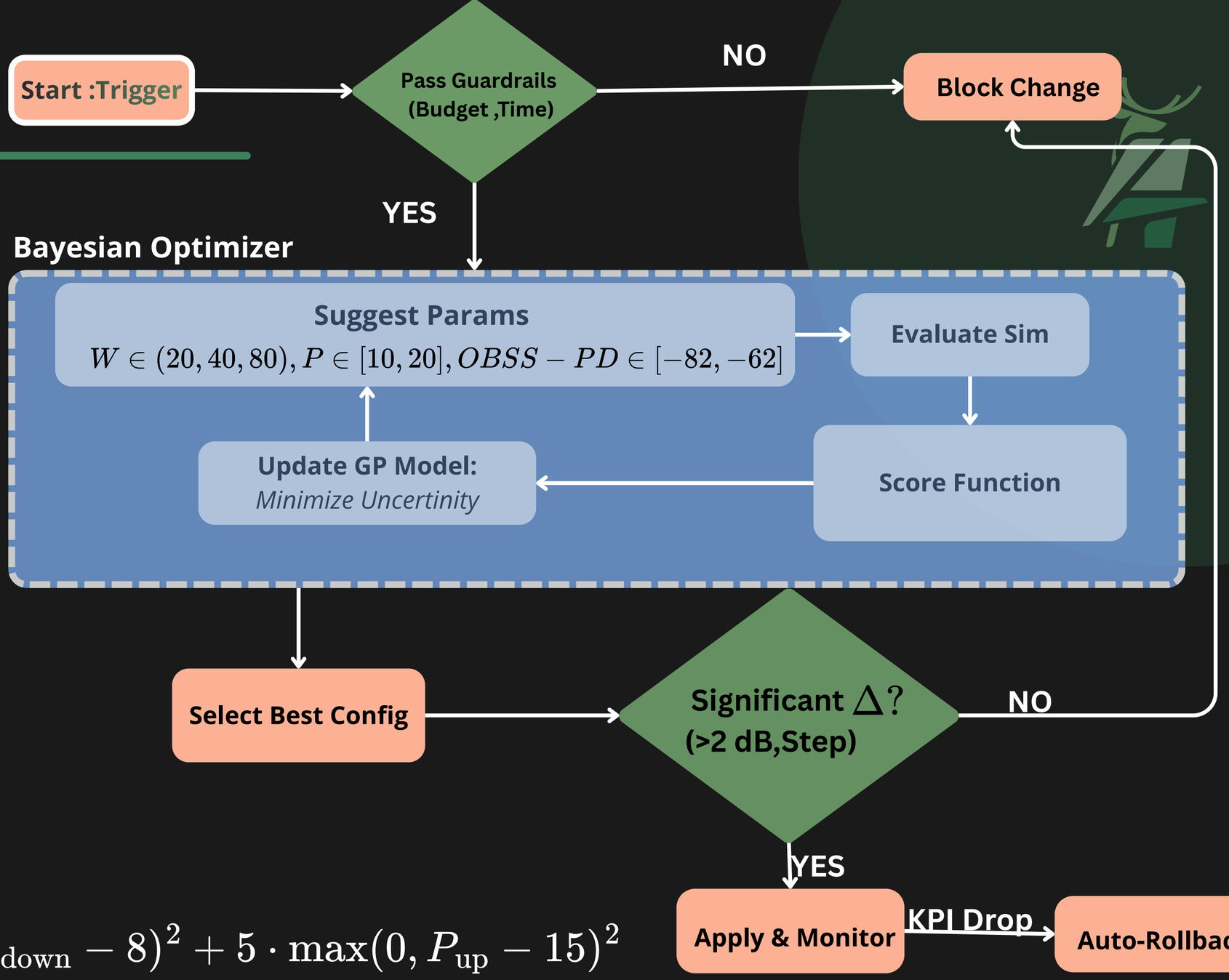
- Ghost network publishes synthetic AP + client RSSI and neighbour interference over MQTT.
- obss_dashboard.py computes per-AP OBSS-PD = min(neighbour+1 dB, weakest client − 5 dB) and animates the interference graph.
- Blue nodes = APs, green/red nodes = strong/bottleneck clients, green dashed edges = spatial reuse possible, red dotted edges = neighbour too loud → blocked.
- Used as a safe playground to tune and verify our OBSS-PD logic before integrating it into the real fast loop.



Safe BO

Self-Optimizing Network Loop

- Proactive Guardrails: Blocks risky changes before execution (e.g., prevents changes during Peak Hours).
- Bayesian Search: Unlike random trial-and-error, it learns.
  - Inputs: Channel Width, Power, OBSS-PD.
  - Goal: Maximize Throughput while keeping Retry Rate < 8 %.
  - Reactive Safety: Automated rollback in < 3 mins if KPIs degrade.



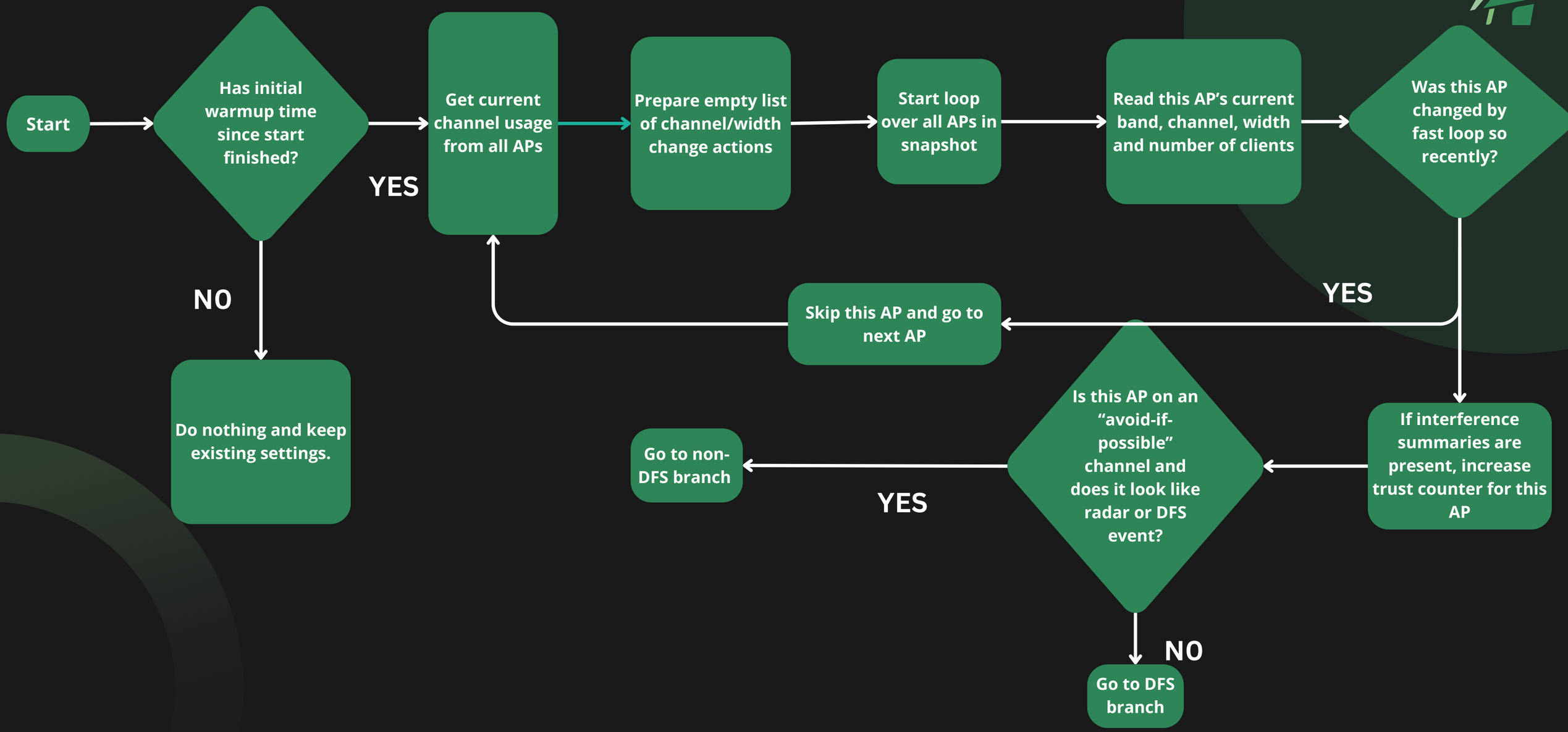
$$\text{Score} = -T_{\text{P50}} + 10 \cdot \max(0, R_{\text{down}} - 8)^2 + 5 \cdot \max(0, P_{\text{up}} - 15)^2$$

BO results with A/B toggle

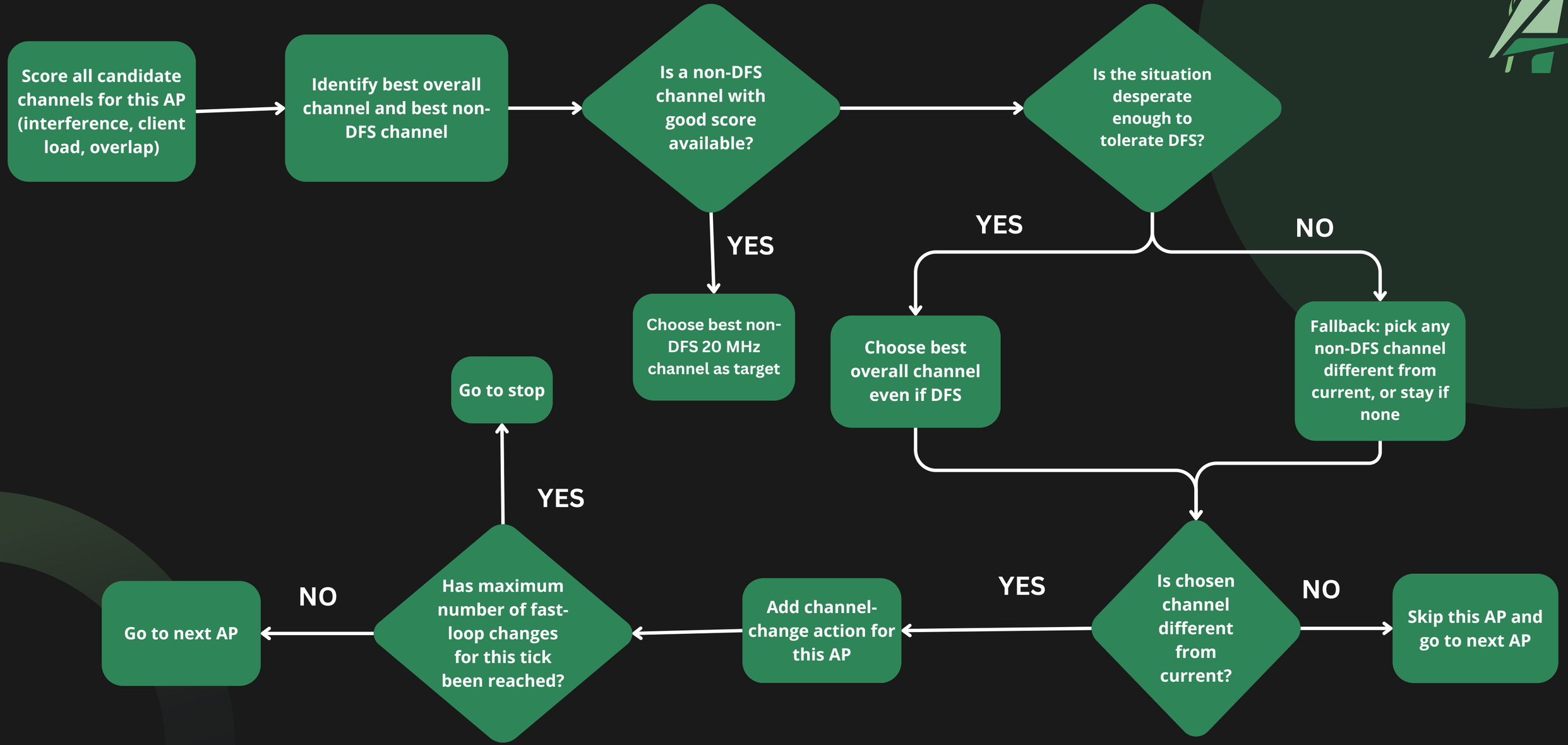
For even APS BO is not used for odd APs BO is applied hence A/B toggle.


```
{
  "ap1": {
    "best": {
      "channel_width": 80,
      "power": 13,
      "obss_pd": -67
    },
    "kpis": {
      "throughput": 144.0,
      "p50_edge_dl": 192.0,
      "retry_down": 0.26907,
      "retry_up": 0.35383385871229783,
      "retry_rate": 0.26907,
      "uplink_per": 0.35383385871229783,
      "clients": 32,
      "qoe_lift": 33.3333,
      "overhead": 0.2
    },
    "used_bo": true,
    "num_clients": 32
  },
  "ap2": {
    "skipped": true,
    "reason": "dfs_unavailable",
    "channel": 112
  },
  "ap3": {
    "best": {
      "channel_width": 80,
      "power": 12,
      "obss_pd": -66
    },
    "kpis": {
      "throughput": 96.0,
      "p50_edge_dl": 192.0,
      "retry_down": 0.2672,
      "retry_up": 0.3554455197387574,
      "retry_rate": 0.2672,
      "uplink_per": 0.3554455197387574,
      "clients": 36,
      "qoe_lift": 166.6667,
      "overhead": 0.2
    },
    "used_bo": true,
    "num_clients": 36
  },
  "ap4": {
    "best": {
      "channel_width": 20,
      "power": 16.0,
      "obss_pd": -78
    },
    "kpis": {
      "throughput": 72.0,
      "p50_edge_dl": 48.0,
      "retry_down": 0.2368,
      "retry_up": 0.3144382036387977,
      "retry_rate": 0.2368,
      "uplink_per": 0.3144382036387977,
      "clients": 27,
      "qoe_lift": 0.0,
      "overhead": 0.0
    },
    "used_bo": false,
    "num_clients": 27
  },
  "ap5": {
    "best": {
      "channel_width": 80,
      "power": 12,
      "obss_pd": -66
    },
    "kpis": {
      "throughput": 96.0,
      "p50_edge_dl": 192.0,
      "retry_down": 0.2697,
      "retry_up": 0.36158293925900864,
      "retry_rate": 0.2697,
      "uplink_per": 0.36158293925900864,
      "clients": 26,
      "qoe_lift": 166.6667,
      "overhead": 0.2
    },
    "used_bo": true,
    "num_clients": 26
  }
}
```

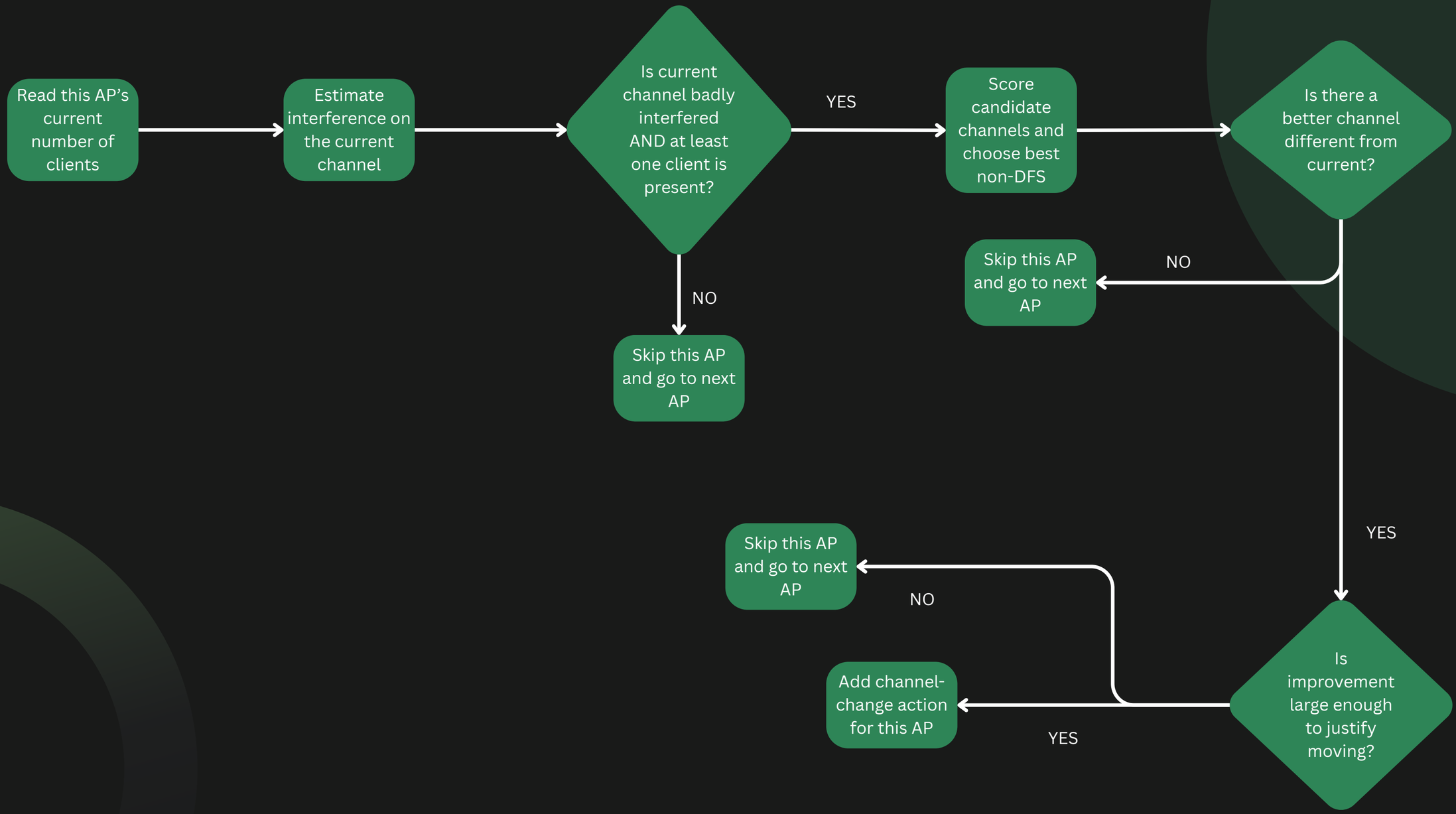
FAST LOOP ARCHITECTURE



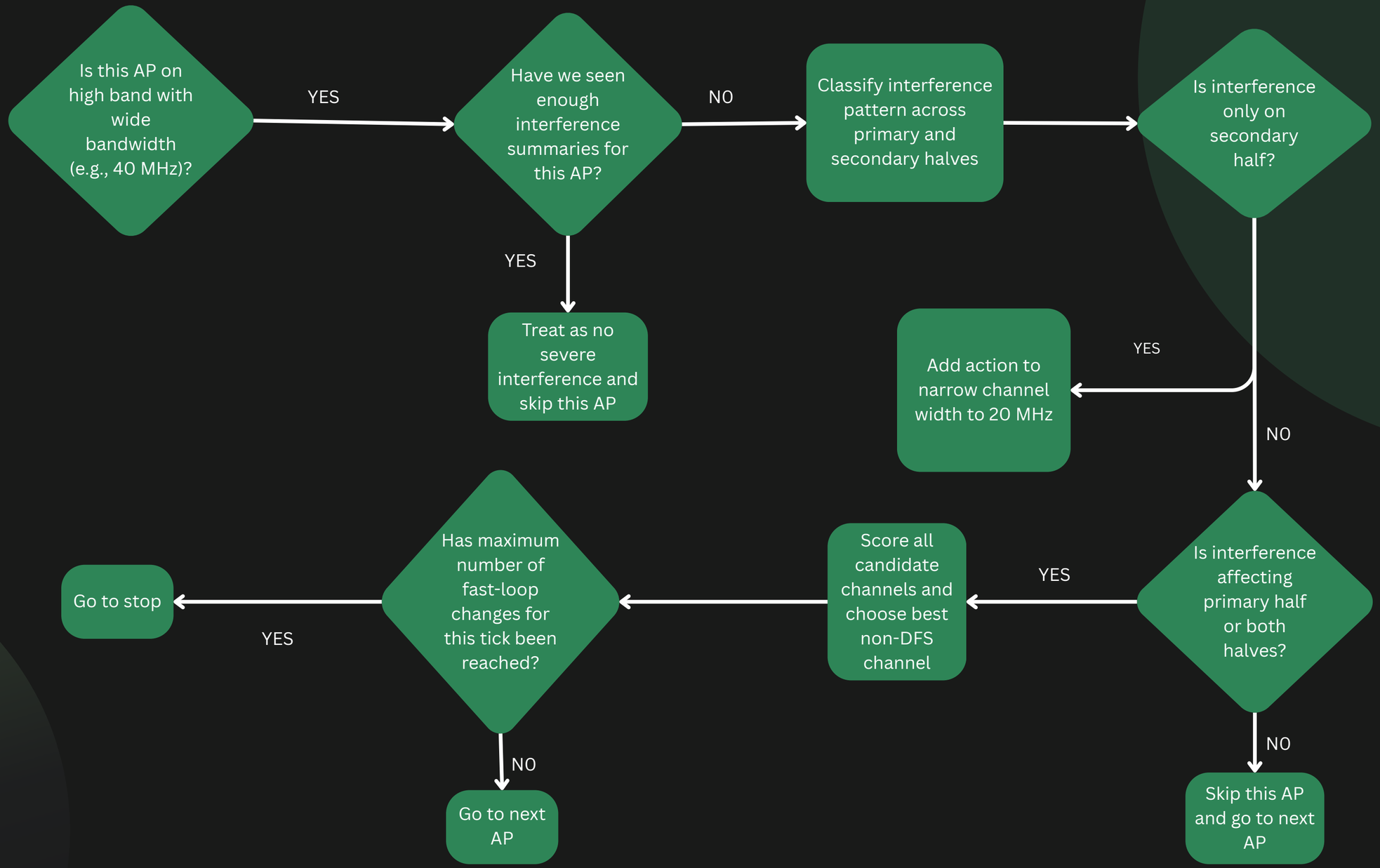
IF A DFS CHANNEL NEED TO BE CHOSEN



IF A NON DFS CHANNEL NEEDS TO BE CHOSEN



CHANNEL WIDTH REDUCTION FOR 40MHZ CHANNELS



EVENT LOOP - EMULATED

The Event Loop activates whenever a category-specific trigger occurs, evaluates the situation and drives the exact behavior defined for that scenario.

Examination center

- **Increase TX power** for stronger, uniform coverage across the exam hall.
- Lock stable RF settings to prevent any channel/width changes during the exam.

Microwave Burst Near Cafeteria

- *Change channel to avoid heavy interference from the cafeteria crowd.*
- *Shrink channel width to maintain stable connectivity in the noisy environment.*

QoE Collapse

- *Decrease channel width to improve stability under congestion.*
- *Increase OBSS-PD and enable aggressive steering to balance the load.*

Manual overrides

- *Freeze the configuration of the affected AP to prevent any automated changes.*
- *Perform an operational check to identify the impacted APs and apply the freeze actions.*

NIGHT MAINTAINANCE WINDOW

- *Allow aggressive tuning since the network is idle and safe for experiments.*
- *Let the RL agent explore fully by relaxing stability constraints during this window.*

EMULATED- CONTROL CONSOLE

Live Controller

Sim date (YYYY-MM-DD): 2025-12-05

Sim time (HH:MM): 09:30

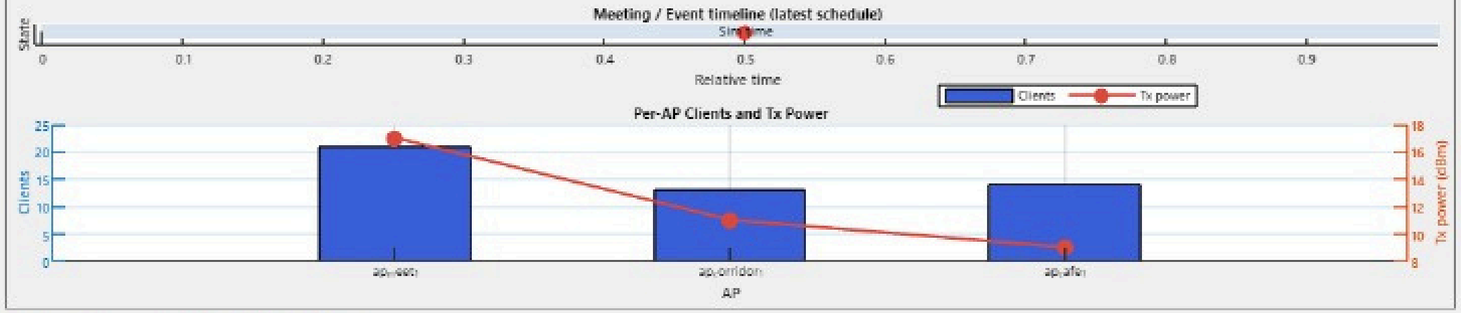
Scenario: meeting_rush

Run control step | Reset simulation state

```
Control step at 2025-12-05 09:30 (meeting_rush).
No meeting schedule active at this time.
Scenario: meeting_rush (busy meeting room).
ap_meet_1 -> ch 11, P=17 dBm, clients=21, interf=0
ap_corridor_1 -> ch 11, P=11 dBm, clients=13, interf=1
ap_cafe_1 -> ch 11, P=9 dBm, clients=14, interf=0
```

AP Snapshot and KPIs

	AP	Role	Channel	Bandwidth	TxPower	Clients	Interference
1	ap_meet_1	Meeting room	11	20	17	21	☐
2	ap_corridor_1	Corridor near meeting	11	20	11	13	☑



Scheduled Profiles (Meeting Quiet Hours)

Meeting date (YYYY-MM-DD): 2025-12-05

Start time (HH:MM): 09:00

End time (HH:MM): 12:00

Meeting room AP IDs: ap_meet_1

Neighbor / corridor / cafeteria...: ap_corridor_1, ap_cafe_1

Options: ☑ Force 20 MHz during meeting ☑ Conservative fast-loop mode

Add Meeting Schedule

Meeting AP power boost (dB):

Neighbor power reduction (dB):

Su	Mo	Tu	We
	1	2	3
7	8	9	10
14	15	16	17
21	22	23	24
28	29	30	31

```
Meeting schedule added for 2025-12-05 09:00-12:00.
Room APs: ap_meet_1
Neighbors: ap_corridor_1, ap_cafe_1
Boost: +3 dB, Neighbor reduction: 3 dB
Force 20 MHz: 1, Conservative fast-loop: 1
```

RF Events and Operational Overrides

RF Events

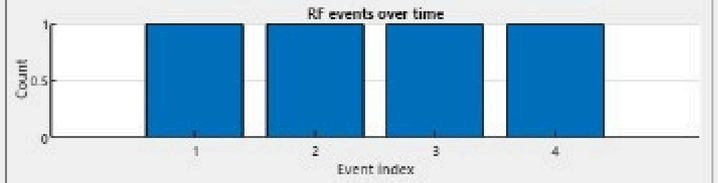
RF event: RF-1 NON_WIFI_BURST

RF affected APs: ap_meet_1

interfered_channel: 1

dfs_channel: 36

delta_obss_pd: 4



	Event	APs	Meta
1	RF-1 NON_WIFI_BURST	ap_meet_1	interfer_ch=1, dfs_ch=36, delta_obss_pd=4
2	RF-1 NON_WIFI_BURST	ap_meet_1	interfer_ch=1, dfs_ch=36, delta_obss_pd=4

```
RF-1 NON_WIFI_BURST on [ap_meet_1]
interfer_ch=1, dfs_ch=36, delta_obss_pd=4
```

Operational Overrides

Operational event: OP-1 MANUAL_OVERRIDE_FREEZE

Affected APs (empty=all): ap_meet_1

Freeze action (enable/d...: enable

Enable flag (OP-2/3/4): ☑ Enabled

power_delta (OP-3): -3

	Time	Event	APs	Meta
1	07:18:20	OP-1 MANUAL_OVERRIDE_FREEZE	ap_meet_1	freeze=enable, enable=1, power_delta=-3
2	07:18:21	OP-1 MANUAL_OVERRIDE_FREEZE	ap_meet_1	freeze=enable, enable=1, power_delta=-3

```
OP-1 MANUAL_OVERRIDE_FREEZE on ap_meet_1
freeze=enable, enable=1, power_delta=-3
```

WHY REINFORCEMENT LEARNING

WiFi RRM is not a one-shot optimization.It is a control decision task.Every change in Channel, Transmit Power, Channel Width,OBSS-PD directly affects future interference,client behavior, and network stability.

WHY OFFLINE LEARNING POLICY

Online Exploration is Unsafe in Live WiFi Networks.Offline RL can learn without touching the live network by leveraging large historical logs available.

WHY CONSERVATIVE Q LEARNING

Problem: Normal Offline RL polices Suffers from Over-Estimation

Conservative Q-Learning explicitly:

- Penalizes Q-values of unknown actions
- Forces the learned policy to stay close to proven safe behavior
- Prefers actions that have worked in real deployments

This is perfect for WiFi networks, where:

- Exploration is dangerous
- Unknown configs can destroy QoE

CQL is robust to dataset shift because it avoids aggressive Q-extrapolation.

Built-In Safety Without External Constraints

MDP FORMULATION

- **State st:** RF snapshot summarised as a graph Gt = (Vt, Et), with AP node features and AP–AP edge features derived from the interference graph.

- **Action at:** joint action across APs, implemented as per-AP discrete actions drawn from A = {NO_OP, POWER_UP, POWER_DOWN, BW_UP, BW_DOWN, OBSS_UP, OBSS_DOWN}.

- **Reward rt:** global QoE reward derived from coverage, throughput, retries, fairness, and configuration churn.

- **Transition st+1:** new telemetry snapshot after the applied actions propagate through traffic dynamics and client mobility.

Reward Engineering

QoE calculation

Rssi Score(edge coverage)
min_rssi = min(rssi)
rt = clip(min_rssi , -90,-40)
rssi score = (rt + 90)/50 (=[0,1]

Retry Score
denote the mean tx retries per client at ap
retries_norm = min(retries/20,1),
retrries_score = 1 - retries_norm

Throughput score
mean phy rate(averaged over tx/rx bitrates at ap i) and max_phy_rate as a reference.
phy_norm = min(avg_phy rate/MAX_PHY_RATE,1)
thr_score = phy_norm * retries_score

QoE = w_rssi*(rssi_score) + w_retries*(retries_score) + w_thr*(thr_score)
Effective_QoE = fairness * mean_QoE

Churn Rate
churn_rate = |aps with changed configurations|÷max(1,|aps|)

Retries Violation
retries violation = max(0.0,(p95 - 8.0) ÷ 8.0)

Final reward = effective_Qoe - lambda_churn*(churn_rate) - lamda_retries*(retries_violation)

Final CQL objective

$$\mathcal{L}_{\mathrm{CQL}}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[\big(Q_\theta(s_t,a_t) - (\mathrm{QoE}_t \cdot \mathrm{Fairness}_t - \lambda_c \cdot \mathrm{ChurnRate}_t - \lambda_r \cdot \mathrm{RetryViolation}_t + \gamma \max_{a'} Q_\theta(s_{t+1},a'))\big)^2\Big] + \alpha\, \mathbb{E}_{(s_t,a_t)\sim D}\Big[\log \sum_a \exp\big(Q_\theta(s_t,a)\big) - Q_\theta(s_t,a_t)\Big],$$

How We Trained and Deployed

First, we constructed a baseline RL policy using one full day of production logs generated under a purely safe, rule-based controller (no rollbacks were required during this phase). We trained our multi-agent controller on this offline dataset using a Conservative Q-Learning (CQL) objective, yielding a stable and safety-aligned initial policy.

Starting from this baseline, we then ran a three-day deployment where the intelligent RRM agents selected actions according to the learned policy, but under strict safety guardrails with automatic rollback on any guardrail violation. At the end of each day, we retrained the agents on the newly collected logs (previous policy + new experience), progressively refining the policy. Over these three days, the agents learned to take increasingly effective actions while remaining within the safety constraints and avoiding guardrail violations.

PreTrain log


```
{"ts": 1764882453.1323392, "step": 2, "phase": "pretrain", "event": "slow_loop_step_done", "data": {"elapsed_sec": 0.10491752624511719,
"sleep_sec": 119.89508247375488, "num_aps": 1, "rl_enabled": false, "cooldown_active": false}}
{"ts": 1764882573.1332018, "step": 3, "phase": "pretrain", "event": "slow_loop_step_done", "data": {"elapsed_sec": 0.0769035816192627,
"sleep_sec": 119.92309641838074, "num_aps": 3, "rl_enabled": false, "cooldown_active": false}}
```

EXPLAINABILITY RESULT

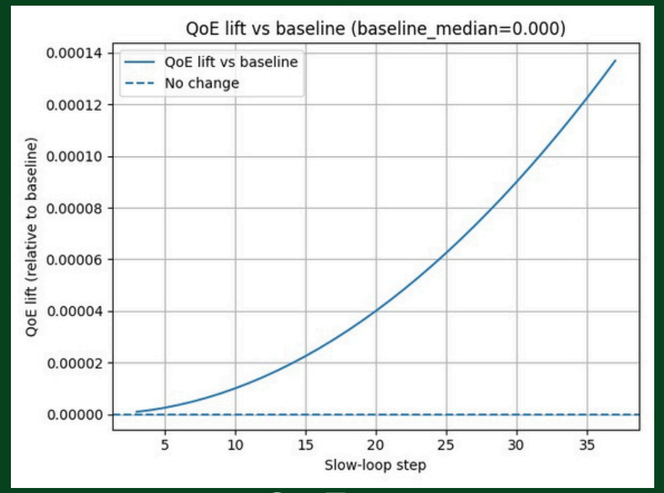
Every slow-loop decision is accompanied by a human-readable explanation

For each AP, the controller records the chosen discrete action, the local metrics and the Q-values for all legal actions.


```
[WHY] {'ap_id': 'ap3', 'action': 'BW_UP', 'q_values': [-10.485315322875977, -10.15392780303955, -18.43416404724121, -10.076863288879395, -20.42693519592285, -10.59659194946289, -15.226102828979492], 'q_advantage_
vs_noop': 0.40845203399658203, 'local_metrics': {'num_clients': 2, 'min_rssi_dbm': -68.0, 'mean_tx_retries': 51.5, 'interference_weight_sum': 0.0}, 'reason_code': 'LOW_INTERFERENCE_WIDEN_CHANNEL', 'reason_text':
'widen channel because interference is low (0.00) and we want more throughput.'}
[WHY] {'ap_id': 'ap1', 'action': 'POWER_UP', 'q_values': [0.5049978494644165, 1.2156533002853394, -9.177584648132324, -0.9009206295013428, -11.581307411193848, -0.7143645882606506, -5.508190631866455], 'q_advanta
ge_vs_noop': 0.7106554508209229, 'local_metrics': {'num_clients': 3, 'min_rssi_dbm': -64.0, 'mean_tx_retries': 2.3333333333333335, 'interference_weight_sum': 0.0}, 'reason_code': 'EDGE_CLIENTS_WEAK_INCREASE_POWER
', 'reason_text': 'increase TX power because edge client RSSI is low (-64.0 dBm) and interference from neighbors is only moderate (0.00).'}
[WHY] {'ap_id': 'ap2', 'action': 'POWER_UP', 'q_values': [0.4797392189502716, 1.145618200302124, -8.874756813049316, -0.952530026435852, -11.063395500183105, -0.47281354665756226, -5.11182165145874], 'q_advantage
_vs_noop': 0.6658789813518524, 'local_metrics': {'num_clients': 4, 'min_rssi_dbm': -51.0, 'mean_tx_retries': 0.0, 'interference_weight_sum': 0.0}, 'reason_code': 'EDGE_CLIENTS_WEAK_INCREASE_POWER', 'reason_text':
 'increase TX power because edge client RSSI is low (-51.0 dBm) and interference from neighbors is only moderate (0.00).'}
```

RESULT LOGS

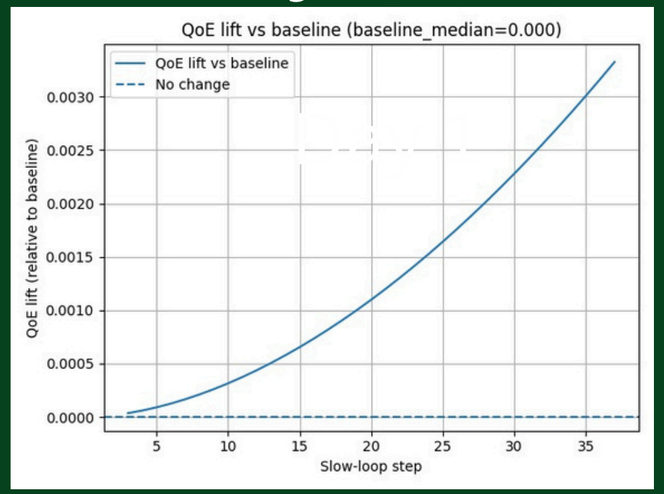
Day 1



QoE

Reward

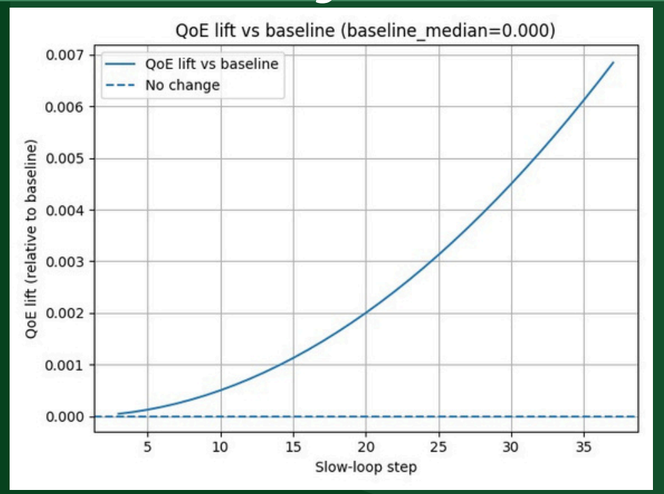
Day 2



QoE

Reward

Day 3

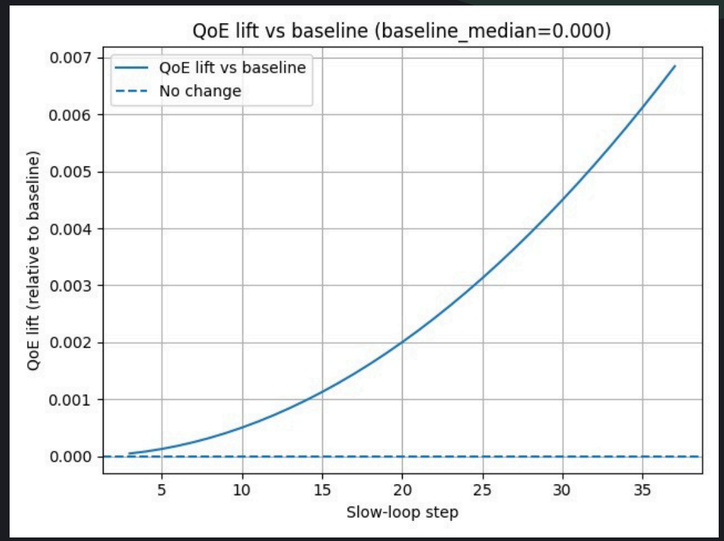


QoE

Reward

CONVERGENCE ANALYSIS & SAFETY VALIDATION

- Day 1 (Exploration): Deep negative spikes confirm active enforcement of Change Budgets (*<0.2 /day*) and Retry Limits.
- Day 3 (Convergence): System learned the "Safe Operating Region," achieving zero config churn.
- Safety Mechanism: The Controller successfully rejected unsafe high-power configurations that violated the P95 Retry SLO (>8%).
- Composite Utility Lift (+0.007): Represents weighted objective (40% Throughput, 30% Retries, 30% RSSI).
- Throughput Conversion: Utility delta translates to ~18% increase in stable Edge Client Throughput.
- Pareto Analysis:
  - Target: Maximize Throughput subject to P95 Retries < 8%.
  - Result: 18% is the Maximum Safe Lift. Pushing >25% triggered reliability violations (as seen in Day 1 logs).



THANK YOU